Multi-objective evolution for car setup optimization

Jorge Muñoz, German Gutierrez, Araceli Sanchis

***Abstract*— This paper describes the winner algorithm of the *Car Setup Optimization Competition* that took place in *EvoStar* (2010). The aim of this competition is to create an optimization algorithm to fine tune the parameters of a car in the The Open Racing Car Simulator (TORCS) video game. There were five participants of the competition plus the two algorithms presented by the organizers (that do not take part in the competition). Our algorithm is a Multi-Objective Evolutionary Algorithm (MOEA) based on the Non-Dominated Sorting Genetic Algorithm (NSGAII) adapted to the constraints of the competition, that focus its fitness function in the lap time. Our results are also compared with other evolutionary algorithms and with the results of the other competition participants.**

I. Introduction

Video games are growing day by day as a popular benchmark to develop and research new algorithms. And with them, competitions are being created to compare the results of different algorithms in different domains. One of the video games that is being used actively during the last years is *The Open Racing Car Simulator* (TORCS). This is a very realistic open source simulator with a sophisticated physics engine that takes into account many aspects as the wheels pressure and their angle with the road, the fuel consumption, the grip of the road, aerodynamic coefficient of the car, the wings angle, collisions, etc. The game provides several different cars, lot of tracks and also tools to create new ones. For these reasons TORCS has being used actively in the last years to create new competitions like the *2010 Simulated Car Racing Championship* [1] [1], *Demolition Derby Competition* [2] or the *Car Setup Optimization Competition* [3] we describe in this paper.

Focusing in the *Car Setup Optimization Competition*, TORCS is a perfectly scenario for an optimization competition due to its sophisticated physics engine. Notice that a physics engine is able to describe what is going to happen given some variables by means of complex equations , but it is usually very difficult to determine which are the appropriate values of the variables that lead to a concrete situation. That is, solving the equations is a really hard problem. Here is where the artificial intelligence comes in developing algorithms to find, not the optimal parameters but the suboptimal parameters which best match with the solution of the problem. Specifically, in the *Car Setup Optimization Competition* the aim is to find the best car setup in order the achieve the maximum distance in a specific period of time. The algorithm developed should be able to find the appropriate wings angles, wheel pressures, gearbox ratios and other parameters that will be explained latter. The optimization algorithms have a maximum of 1000000 game ticks, around five hours and half of time TORCS, that can use to simulate. This means that each algorithm must decide how much time spends in each simulation to check the parameters and get the fitness value. Each simulation starts where the last one ends, so there are not two simulations equal and this leads to new problems that need to be solve. For example, if the simulation time is too short the car only run in a small portion of the track and the parameters are not generalize for a complete lap, but if the simulation time is too long the algorithm will only check very few combinations of parameters.

Fig. 1. Screenshot of TORCS

We decided to use a *Multi-Objective Evolutionary Algorithm* (MOEA) for the competition because of two main reasons. The first one is that with a multi-objective algorithm we can improve the individuals in more than one objective, for example we can reduce the lap time at the same time we increase the top speed and decrease the damage. And the second one, is because we observed that a multi-objective algorithm converges faster in the first generations than other evolutionary algorithms like a single genetic algorithm, even that ones that combine more than one objective in the fitness [2]. Due to the simulation time is not long enough to run a proper number of generations to get a good individual the convergence in the first generations is an important aspect to bear in mind.

The next section, Section II, covers a detailed description of the competition. In Section III the algorithms we used in the competition are described. Next, Section IV, the results

J. Muñoz, G. Gutierrez, A. Sanchis are with the Computer Science Department, Universidad Carlos III de Madrid, Avda. de la Universidad 30, 28911 Leganés, Spain (emails: { jmfuente, ggutierr, masm }@inf.uc3m.es).

[1] http://cig.dei.polimi.it/?page_id=134

[2] http://www.coboslab.psychologie.uni-wuerzburg.de/competitions/

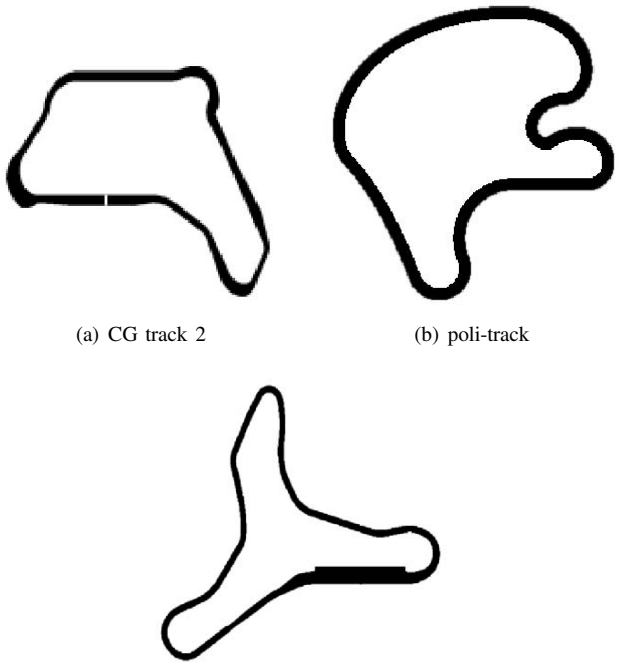
[3] http://cig.dei.polimi.it/?page_id=103

of our evolutionary algorithms are shown and compared with the results of the competition. Finally, in Section V, we expose the conclusions and the future works.

II. Car Setup Optimization Competition

A complete description of the *Car Setup Optimization Competition* is located in [3]. We describe in this section the most important aspects of this competition in order to understand the algorithms we describe in Section III.

In the competition the algorithm proposed has to find the best car setup for three tracks. These tracks were unknown for the participants, we know now that these three tracks were *poli-track*, *dirt 3* and *CG track 3* (see Figure 2) For each track, the competition is divided in two phases: optimization and evaluation. In the optimization phase the optimization algorithms proposed runs during one million of game ticks, around five and a half hours of game time simulation, and sent to the server the best parameters it finds. During the evaluation the best car setup found is evaluated during 10000 game ticks and the distance covered in that time is the score of the algorithm. The higher this value is the better it is.



(a) CG track 2 (b) poli-track

(c) dirt 3

Fig. 2. Tracks used in the car setup optimization competition

The parameters the algorithm has to optimize are 22:

- gearbox ratios (5 parameters)
- angle of front and rear wing (2 parameters)
- brake system (4 parameters)
- front and rear anti-roll bar (2 parameters)
- wheels camber, ride height and toe (5 parameters)
- suspension course and spring (4 parameters)

These parameters are represented in the system as a vector of real numbers in the interval [0,1]. The parameters suffers a random mapping for each execution, there is no fixed position of each parameter in the vector. Therefore, the optimization algorithm can not assume any prior knowledge about the parameters to optimize.

The competition software provides an application program interface (API) that allows the algorithms to evaluate a parameters vector. The algorithm sends the parameters to the game server and the number of game ticks of the simulation, then the game server returns to the algorithm the best lap time, the top speed, the distance raced and the damage suffered.

III. Optimization algorithms

In this section we describe the two algorithms we tested before the competition. The first one is a new evolutionary algorithm developed from the concepts of the genetic algorithms and ideas from the artificial immune systems [4], [5]. The second one is a multi-objective algorithm based on the Non-Dominated Sorting Genetic Algorithm (NSGA-II) [6]. For both algorithms one individual of the population is the parameters vector we mentioned before and the initial population is randomly created with a normal distribution.

Before we explain the algorithms we are going to explain first the common policy followed to determine the simulation time for each evaluation.

A. Simulation time

One important aspect of the competition is the amount of simulation time. We need to decrease as much as possible this value to increase the number of evaluations the algorithm can perfom. But, we have to bear in mind that the simulation time have to be longer enough to get proper values from the simulation. We think that a good value for the simulation is the one that allows the car complete a lap. Although the objective in the competition is to reach the biggest distance in a fixed amount of time, this simulation time is longer than a lap in the most of the tracks, so we have to reduce the lap time as much as possible. We also know that in the first evaluations the lap time is going to be longer than in the last evaluations, with better car setups. So we need a simulation time that also evolves with the car setups, starting long and decreasing at the same time the cars improve their setup. We set the simulation time as:

$$\text{simulation\_ticks} = \frac{\alpha \cdot \text{best\_lap\_time}}{\text{rate\_simulation\_tick}} \quad (1)$$

where α is a number close to 2 , *best_lap_time* is the best lap time ever and *rate_simulation_tick* is the relation between a game tick and a second. The reason why α must be close to 2 is because not two consecutive evaluations start in the same point of the track, indeed, each evaluation starts in the point of the track where the last evaluation ends. So if a simulation starts just after the car pass the start line it needs almost 2 laps to get one proper lap time. So, to avoid to lose setups faster than our current best setup we have to allow the car completes a lap regardless of in which point of the track

the simulation starts. We set α to 1.7 for the competition because we observed better results than with other values.

B. Evolutionary Algorithm (EA)

The evolutionary algorithm we developed is a genetic algorithm with some ideas from the artificial immune systems [4], [5]. We started with the Artificial Immune Evolutionary Algorithm (AIEA) used in [7] for a constraint satisfaction problem, and added crossover. So, instead of applying mutation in all the generations we used randomly mutation and crossover. We presented this algorithm in the *Car Setup Optimization Competition* in 2009. The pseudo-code of the final algorithm is described in Algorithm 1.

Algorithm 1 Evolutionary Algorithm

Require: population size, tournament size, stop criterion, crossover operator

1: generate the initial $population$
2: **for** each $individual$ in $population$ **do**
3: calculate fitness of the $individual$
4: **end for**
5: $maxfit \Leftarrow$ maximum fitness in the $population$
6: $minfit \Leftarrow$ minimum fitness in the $population$
7: **repeat**
8: **if** randomly with probability 0.5 **then**
9: {mutation:}
10: $individual\_to\_clone \Leftarrow$ select the best of a tournament
11: $ind \Leftarrow$ clone $individual\_to\_clone$
12: $fit \Leftarrow ind \rightarrow fitness$
13: $mutation\_range \Leftarrow \frac{fit-minfit}{maxfit-minfit}$
14: mutate ind proportional to $mutationRange$
15: **else**
16: {crossover:}
17: $parent\_1 \Leftarrow$ select the best of a tournament
18: $parent\_2 \Leftarrow$ select the best of a tournament
19: $ind \Leftarrow$ crossover $parent\_1$ and $parent\_2$
20: **end if**
21: $individual\_to\_remove \Leftarrow$ select the worst of a tournament
22: remove $individual\_to\_remove$ from $population$
23: add ind to $population$
24: $maxfit \Leftarrow$ maximum fitness in the $population$
25: $minfit \Leftarrow$ minimum fitness in the $population$
26: **until** stop criterion

The mutation operator selects between 1 and 3 positions in the parameters vector and modify those numbers an amount that is variable between 0.001 and 0.2. When we say in Algorithm 1 the individual is mutated proportional to the $mutationRange$ (line 14) this means that we apply between 10 and 100 times the mutation operator to the same individual proportional to the $mutationRange$ value. The better the individual is, the less times we apply mutation to that individual. Good individuals relatively to the population will suffer a small mutation, small variation in their chromosome, while bad individuals will suffer a big mutation, big difference in their chromosomes..

As crossover for the algorithm we used a very simple one that only mix the values of the vectors of the parents. First two parents are selected by tournament of size 3, then each position of the vector of parameters in the offspring is filled with the value of that parameter in one of the parents. The parent is choosen randomly with the same probability for each position in the vector.

We use as fitness the lap time (note that the algorithm tries to minimize the fitness value). As we said before this is for us the most important value we get from the simulation. When the controller does not perform a complete lap (cross the start line twice) we do not have a lap time, which happens frequently in the first evaluations. So, we set as lap time the simulation time multiplied by 10 plus a value that depends on the top speed and other that depends of the distance covered, as in Equation 4 (β is equals to 0.2 to give more relevance to the speed value than to the distance covered value). With this fitness we give more relevance to the lap time and when we do not have lap time we give more relevance to the distance covered and the top speed.

$$\text{speed\_value} = 1 - \frac{\text{top\_speed}}{\text{maximum\_speed}} \tag{2}$$

$$\text{distance\_value} = 1 - \frac{\text{distance\_raced}}{\text{maximum\_distance}} \tag{3}$$

$$\begin{aligned}\text{fitness} = {} & \text{best\_lap\_time} \cdot 10 + \beta \cdot \text{speed\_value} \\ & + (1-\beta) \cdot \text{distance\_value}\end{aligned} \tag{4}$$

In Equation 2 top_speed is the maximum speed the car reach in the evaluation while maximum_speed is the maximum speed the car can reach. In Equation 3 distance_raced is the distanced covered by the car during the simulation while maximum_distance is the distanced the car will cover if it would drive at the maximum speed.

The damage is taken into account, when it is greater than 10 we multiply the fitness by 1.1 to increase it, and penalize these individuals.

C. Multi-Objective Evolutionary Algorithm (MOEA)

The multi-objective algorithm developed is an adaptation of NSGA-II [6]. NSGA-II is an algorithm created with the ability to find multiple solutions in the pareto front that have good results in different objectives. This is done by the concept of dominance and the crowding distance.

The crossover and mutation operators are the same as in the evolutionary algorithm we explained before. The objectives used for the evaluation of each individual are four, all of them normalized between 0 and 1 where 0 is the best possible value and 1 the worst. The objectives are:

- the lap time (see Equation 5)
- the top speed (see Equation 6)
- the distanced raced (see Equation 7)
- the damage (see Equation 8)

$$\text{objective\_1} = \frac{\text{lap\_time}}{\text{simulation\_ticks} \cdot \text{rate\_simulation\_tick}} \quad (5)$$

$$\text{objective\_2} = 1 - \frac{\text{top\_speed}}{\text{maximum\_speed}} \quad (6)$$

$$\text{objective\_3} = 1 - \frac{\text{distance\_raced}}{\text{maximum\_distance}} \quad (7)$$

$$\text{objective\_4} = 1 - \frac{1}{\frac{\text{damage}}{10} + 1} \quad (8)$$

The aim to use a multi-objective algorithm is to increase the convergence of the algorithm in the first generations. As we said before the total simulation time we can use is not enough for a big number of generations, it is around 6 o 7 generations with populations of 30 individuals in an average length track. The idea is not to lose individuals that can reach very high speeds or suffer very few damage while the algorithm is trying to find the best car setup to complete a lap in the less possible time. If you cross an individual with high top speed and other with a normal lap time you can get a very fast driver, much more than if you only take into account the lap times and you cross two individuals with normal lap times.

IV. Results

All the experiments were run 10 times per each track and the average distance covered in 10000 game tics is the score of the algorithms. The higher this value is the better.

In the competition 3 tracks were used: *poli-track*, *dirt 3* and *CG track 3* (see Figure 2) . As *poli-track* is not available in the game we omit this track for our experiments, although this track is used in the competition.

The experiments of our algorithms were run after the competition with the idea of comparing our results with the competition results.

A. Algorithms results

For both evolutionary algorithms we use populations of 30 individuals. Table I shows the results of the algorithms, EA and MOEA, for two of the tracks. This is the average distance covered in 10000 game ticks for 10 runs. The standard deviation (Std. Dev.) is also shown.

In the first track, *CG track 2*, the MOEA gets better results, even the standard deviation is smaller than in the other algorithm. However in the second track, *Dirt-3*, there is no difference in both algorithms, although MOEA still gets better standard deviation. The reason why this happens in the second track is because this tracks has wall in the edgeds and the cars never go out of the track. So the evolution process leads the most of the times to create faster car setups that nevers steer and are always crashing with these walls. This is less frequently in the MOEA because of the objective that take into account the damage.

B. Competition results

Table II and Table III shows the distance covered in 10000 game ticks. Table II shows the results the participants achieve in average for 10 runs in each of the three tracks while Table III show the results the organizers achieve with their own algorithms.

We do not know all the algorithms used by the other participants, there is not any publication related with them yet. Two of them used particle swarm optimization (Garcia/Saez-PSO and Walz-PSO) other an evolutionary algorithm (Munoz/Martin/Saez-EA) we do not know and the last one is completely unknown (Fernandez/Cotta/Fuentes). Our MOEA algorithm is *Munoz-MOEA*. For each track the winner was a different algorithm. In *CG track 2* our MOEA was the best algorithm with a difference of more than 14% of the distance raced over the second best result. While in the other two tracks where our algorithm is not the best the difference between our algorithm and the best one is less than 4% of the maximum distance raced.

If we compare the results of the competition (Table II) with our previous results (Table I), we can see as our previous results are better than in the competition. This does not mean any result is wrong but the process we follow to get the results, using the API we have, maybe diffiers in the process followed in the competition [4]. The reason lead us to think this is that the difference is around 8% of the maximum distance covered in both common tracks.

If we compare our MOEA with the algorithms developed by the organizers of the competition (see Table III) we can see that the difference with a simple genetic algorithm (Cardamone-SimpleGA) is not really significant. Our algorithm reach better distances in all the tracks but the difference rounds the 4% of the total distance raced. But if we compare MOEA with CMA-ES [8] our algorithm is worse for all the tracks, with a difference of the 5% of the total distance raced. The *Kemmerling-CMA_ES* organizer is clearly the best algorithm of all.

V. Conclusions and future work

We are glad to say that the results in the competition were better than we expected. The multi-objective algorithm gets better results than the other algorithms presented in the competition. However, when we compare our algorithm with a simple genetic algorithm developed by one of the organizers the difference does not look really significant. We have to notice that this simple genetic algorithm is also better than the most participants of the competition in two of the tracks. If we compare our best algorithm with CMA-ES [8], the other algorithm presented by the organizers, our results are worse in all the tracks. This means that there is still more room for improvement in the algorithms.

CMA-ES is an evolutionary strategy based on the covariance matrix adaptation that looks like has a fast convergence to good solutions in the first steps of the evolutionary

[4] there is not any technical report that detailed the process follow to run the experiments in the competition

TABLE I

RESULTS OF OUR TWO EVOLUTIONARY ALGORITHMS

	CG track 2		Dirt-3		Overall
Algorithm	Distance	Std.Dev.	Distance	Std.Dev.	Distance
EA	10202.16	916.56	6572.12	347.54	16774.28
MOEA	10932.40	429.61	6572.44	272.37	17504.85

TABLE II

RESULTS OF THE CAR SETUP OPTIMIZATION COMPETITION

Competitor	CG track 2	Poli-track	Dirt-3	Overall
Munoz-MOEA	**9831.83**	7654.01	6128.29	**23614.13**
Garcia/Saez-PSO	8386.77	**7979.86**	5021.41	21388.04
Walz-PSO	8408.35	7304.54	5336.88	21049.77
Fernandez/Cotta/Fuentes	7553.21	5931.47	**6263.40**	19748.08
Munoz/Martin/Saez-EA	8167.60	7718.36	4629.33	20515.29

TABLE III

ORGANIZERS RESULTS

Organizer	CG track 2	Poli-track	Dirt-3	Overall
Cardamone-SimpleGA	9563.08	7273.06	5932.09	22768.23
Kemmerling-CMA_ES	10410.13	8392.49	6415.87	**25218.49**

process. And, although this algorithms has beat the rest of the algorithms presented in this paper, we still believe that a multi-objective algorithm is the best one to get the best results in a domain like the *Car Setup Optimization Competition*. There is variant of CMA-ES for multi-objective problems called MO-CMA-ES [9] which is our next line of research for the competition, with a lot of possibilities to improve the results of this year.

ACKNOWLEDGMENT

This work was supported in part by the University Carlos III of Madrid under grant PIF UC3M01-0809 and by the Ministry of Science and Innovation under project TRA2007-67374-C02-02.

REFERENCES

[1] D. Loiacono, J. Togelius, P. Lanzi, L. Kinnaird-Heether, S. Lucas, M. Simmerson, D. Perez, R. Reynolds, and Y. Saez, "The WCCI 2008 simulated car racing competition," in *Computational Intelligence and Games, 2008. CIG '09. IEEE Symposium On*, Dec. 2008, pp. 119–126.

[2] J. Muñoz, G. Gutierrez, and A. Sanchis, "Evolutionary genetic algorithms in a constraint satisfaction problem: Puzzle eternity ii," in *Proceedings 10th International Work-Conference on Artificial Neural Networks*, June 2009, pp. 220–227.

[3] L. Cardamone, D. Loiacono, and P. L. Lanzi, "Car Setup Optimization. Competition Software Manual," Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy, Tech. Rep., 2010.1.

[4] L. de Castro and F. Von Zuben, "Artificial Immune Systems: Part I– Basic Theory and Applications," Universidade Estadual de Campinas, Dezembro de, Tech. Rep., 1999.

[5] L. de Castro and F. V. Zuben, "Artificial Immune Systems: Part II A Survey Of Applications," Department of Computer Engineering and Industrial, Tech. Rep., 2000.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[7] J. Muñoz, G. Gutierrez, and A. Sanchis, "Controller for torcs created by imitation," in *IEEE Symposium on Computational Intelligence and Games.*, September 2009, pp. 271–278.

[8] N. Hansen, "The CMA evolution strategy: a comparing review," *Towards a new evolutionary computation*, pp. 75–102, 2006.

[9] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *Evolutionary Computation*, vol. 15, no. 1, pp. 1–28, 2007.